## AMENDMENT TO THE SPECIFICATION

Please amend the specification by marked up replacement paragraph(s) as follows.

Please replace paragraph 23 as follows:

**[23]** Each client 101, 103, and 105 has initiated a session with a corresponding instance of Java virtual machine 111, 113, and 115, which are capable of executing Java programs on the web server 100. The web server 100 also stores various web and server pages, including server page 131, which contains both static markup text and coded fragments embedded among the static markup text. By way of example, FIG. 1 shows what can be the contents 133 of ~~an~~ a server page 313. In this example, the server page contents 133 includes both static HTML text and, in bold, embedded Java code between the delimiters <% and %>. The web server also stores the preloaded markup text 141 of the server page 131 in a shared memory accessible to different users, sessions, and processes.

Please replace paragraph 26 as follows:

**[26]** When the servlet class is first used by any of the servlet instances 121, 123, and 125, the static class initializer code is executed (step 207). The static class initializer, when executed, reads each ~~of~~ line of the resource file and initializes the corresponding static character array with the read line. At step 209, the class with its static character arrays being initialized from the resource file is then "hotloaded" into shared memory for use by any session. Implementation details of hotloading are described in the following section.

Please replace paragraph 30 as follows:

**[30]** Database system 300 comprises, among other components, a database memory 301 for storing information useful for processing calls and a number of server processes 303 and 305 for handling individual calls. These server ~~process~~ processes 303 and 305 may involve initiating an instance of a Java Virtual Machine to perform actions programmed in Java. The database memory 301 includes various memory areas used to store data used by server processes 303 and 305. These memory areas include a database instance memory 310, session memories 311, 313, 315, and 317, and call memories 321 and 323. It is to be understood that the number of the session memories and call memories in FIG. 3 is merely illustrative and, in fact, the number of such memories will vary over time as various clients make various calls to the database system 300.

Please replace paragraphs 34-35 as follows:

**[34]** At any given time, a server process is assigned to process a call submitted by a single client. After the server process completes its processing of a call from one client, the server process is free to be assigned to respond to the call of another client. Thus, over a period of time, a server process may be assigned to process calls from multiple clients, and a client may use multiple server processes to ~~handles~~ handle its various calls. At any given time, the number of calls requiring execution by a server process is typically much fewer than the current number of active clients. Thus, database system 300 is typically configured to execute fewer server processes than the maximum number of active clients.

**[35]** Some static class variables that are good candidates for sharing because they are not modified after construction ~~but~~ still require execution of a static class initializer to be properly constructed. One example includes initializing the static class variables of a servlet to include

the static markup text that was saved in a resource file. FIG. 4 is a flowchart for a method of managing a run-time environment, carried out during construction of objects for later loading to a shared read-only memory in a server of the computer system.

Please replace paragraphs 38-40 as follows:

**[38]** The class along with the initialized static class variable, in step 405, are then saved in a second class file that shadows the class file, with a name related to that of the class file but with a different extension, path, or other filename component. The second class file with the saved, constructed class object is loaded into a globally shared memory in step 407, such as a shared read-only memory. Thus, the values of constructed static class objects can be placed into the globally shared memory via the second class file.

**[39]** ~~Alternative~~ Alternatively a specific class can designated, for example by a system administrator ~~specifies~~ specification, for migration to shared read-only memory. The designated class, if not already in shared memory, is then migrated into shared memory and is stored in the database. Later, the class is loaded from the database into the shared memory by looking up the name of the class in the database. The designation and migration may be performed in one process by the system administrator, and the loading may be performed in one or more subsequent processes by other users, thereby saving the overhead involved in constructing the object.

**[40]** Further details of the implementation and use of this mechanism may be found in the commonly assigned, co-pending patent application, serial no. 09/512,618, entitled "Method and apparatus for managing shared memory in a run-time environment," filed on Feb. 25, 2000 by Harlan Sexton et al., now U.S. Patent No. 6,829,761, the contents of which are hereby incorporated by reference in their entirety.